ETH zürich

ETH Library

OpenTIPE: An Open-source Translation Framework for Interactive Post-Editing Research

Conference Paper

Author(s):
Landwehr, Fabian; Steinmann, Thomas; Mascarell, Laura

Publication date:
2023-07

Permanent link:
https://doi.org/10.3929/ethz-b-000613497

Rights / license:
Creative Commons Attribution 4.0 International

Originally published in:
https://doi.org/10.18653/v1/2023.acl-demo.19

This page was generated automatically upon download from the ETH Zurich Research Collection.
For more information, please consult the Terms of use.

OpenTIPE: An Open-source Translation Framework for Interactive Post-Editing Research

Fabian Landwehr and **Thomas Steinmann** and **Laura Mascarell**
ETH Zurich
{fabian.landwehr,thomas.steinmann,lmascarell}@inf.ethz.ch

Abstract

Despite the latest improvements on machine translation, professional translators still must review and post-edit the automatic output to ensure high-quality translations. The research on automating this process lacks an interactive post-editing environment implemented for this purpose; therefore, current approaches do not consider the human interactions that occur in real post-editing scenarios. To address this issue, we present OpenTIPE, a flexible and extensible framework that aims at supporting research on interactive post-editing. Specifically, the interactive environment of OpenTIPE allows researchers to explore human-centered approaches for the post-editing task. We release the OpenTIPE source code[1] and showcase its main functionalities with a demonstration video[2] and an online live demo.[3]

1 Introduction

Recent advances in Machine Translation (MT) and the adoption of neural architectures (Bentivogli et al., 2016) led to significant improvements in translation quality aspects such as fluency or adequacy (Bentivogli et al., 2016; Bojar et al., 2017). Despite these advancements, MT is not yet on a par with human performance (Läubli et al., 2018; Toral, 2020) and human post-editors need to edit the MT output to obtain high-quality translations that also adapt to a specific domain and style.

To reduce the manual efforts, the research community proposed to automate this process and implemented Automatic Post-Editing (APE) models that automatically learn post-editing rules from revised translations (do Carmo et al., 2020). The use of these models is specially indicated in environments where the MT system that generates the translations is not accessible for retraining (Chatterjee et al., 2015). To date, the automatic corrections generated by the state-of-the-art APE models still require proofreading, so there is no solution that fully automates the translation process. In fact, the post-editing task remains mostly manual in production workflows.

Instead of fully automating the post-editing task, Escribe and Mitkov (2021) suggest that post-editing would greatly benefit from human-centered approaches that leverage the post-editor's corrections and their interactions with the translation interface. For example, APE models could improve over time by incrementally learning from human corrections (Chatterjee et al., 2017).

While human-computer interaction has been explored in MT with the help of translation frameworks such as CASMACAT (Alabau et al., 2013), there is no such interactive environment designed for the post-editing task (do Carmo et al., 2020; Escribe and Mitkov, 2021). Therefore, current research in Interactive Post-Editing (IPE) is limited to simulate the human interaction by feeding pre-existing corrections to the post-editing process sequentially (Ortiz-Martínez and Casacuberta, 2014; Chatterjee et al., 2017; Negri et al., 2018a). Additionally, human corrections are scarce and these approaches often rely on synthetic post-edited datasets such as eSCAPE (Negri et al., 2018b). Although these artificial settings enabled valuable research in this field, they lack the human intervention as in real-world post-editing scenarios.

In this paper, we present OpenTIPE, an open-source framework that enables research on post-editing in an interactive environment. In particular, OpenTIPE implements the following main features:

- **Easy-to-use interface** that allows users to automatically translate a text and post-edit it. To support the user during the post-editing process, the tool provides **automatic post-editing suggestions** from an APE model, which can be directly applied to the revised translation (see Section 3.1).

[1] Link to GitHub repository.
[2] https://youtu.be/G3Hb8_hnKIk
[3] https://www.opentipe-demo.com

- **Collection of human data**, such as user corrections and post-editing feedback (e.g. whether the automatic suggestions were applied). The collected data is a valuable resource to incrementally improve APE models in a continuous feedback loop (Section 4).
- **Logging of post-editing activity**, such as the user inactivity, and the time at the start and end of the post-editing task. The implemented logging allows researchers to measure post-editing efforts and evaluate the post-editor experience on different settings (Section 4).
- **Modular and extensible** microservice architecture using Docker containers,[4] which facilitates the extension and implementation of additional services (e.g. translation or APE models) and features (e.g. new logging activity or user interface design). Section 3 describes the OpenTIPE architecture in detail.

To the best of our knowledge, this is the first interactive environment designed to facilitate research on IPE. We hope that OpenTIPE fosters further research in this field that can be applied to improve the overall post-editing experience.

2 Related Work

Most of the related work focuses on implementing APE approaches to automate the post-editing task. However, these approaches do not enhance human-computer interaction. In fact, human-centered approaches have been only explored in MT settings. In this section, we first summarize the work on APE and their online approaches, which simulate the post-editor behaviour by implementing continuous streams of corrections. We then describe the research on interactive MT, which is the most inline with this work.

Automatic Post-editing The annual WMT conference[5] has been hosting shared tasks on APE since 2015, going through both statistical and neural MT eras (Junczys-Dowmunt, 2018; do Carmo et al., 2020). An important finding is that the performance of APE models are highly influenced by the quality of the MT output, hence improving neural MT translations is particularly challenging for these models. Additionally, automatic metrics such as TER (Snover et al., 2006) and BLEU (Papineni et al., 2002) cannot always reflect the improvements of the APE models, and researchers need to conduct manual evaluations to gain insights on the quality of the APE output (Akhbardeh et al., 2021). Nevertheless, APE models are an essential component in translation workflows, where the MT system is used as a black box and its parameters are not available for retraining. In this setting, it would be beneficial to leverage human feedback to gradually improve the performance of the APE model in place.

Online APE To simulate human post-editions, APE models apply online learning methods that feed continuous streams of data to the model. While online methods have been previously adopted in phrase-based APE (Ortiz-Martínez and Casacuberta, 2014; Simard and Foster, 2013; Chatterjee et al., 2017), only Negri et al. (2018a) apply online learning to neural APE. Specifically, Negri et al. (2018a) iterate over a pre-existing dataset of corrections, updating the model parameters on the fly for every instance. Similar works address online learning in neural MT. For example, Kothur et al. (2018) update the model parameters one sentence at a time as in Negri et al. (2018a). In contrast, other approaches avoid updating the model parameters and retrieve sentences with similar contexts from a translation memory during decoding (Gu et al., 2018; Wang et al., 2021).

Interactive MT In professional translation environments, human experts benefit from using computer-assisted translation technologies (i.e. CAT tools). For example, translation memories store previously-approved translations, so they can be reused later on. To further investigate the human-computer interaction in translation workflows, researchers proposed several frameworks for phrase-based MT (e.g. Transtype2 (Esteban et al., 2004) and CASMACAT (Alabau et al., 2013)) and neural MT (Knowles and Koehn, 2016; Santy et al., 2019). However, these technologies are not optimal to investigate human interaction in post-editing, and therefore there is a lack of research in this area. For example, CASMACAT offers alternative translation suggestions that come from an MT system, whereas our work integrates the output of an APE model. Similarly to the prior work in interactive MT, we implement automatic logging strategies to collect user interactions during the post-editing process for further analyses.

[4] https://www.docker.com
[5] https://www.statmt.org/wmt22/

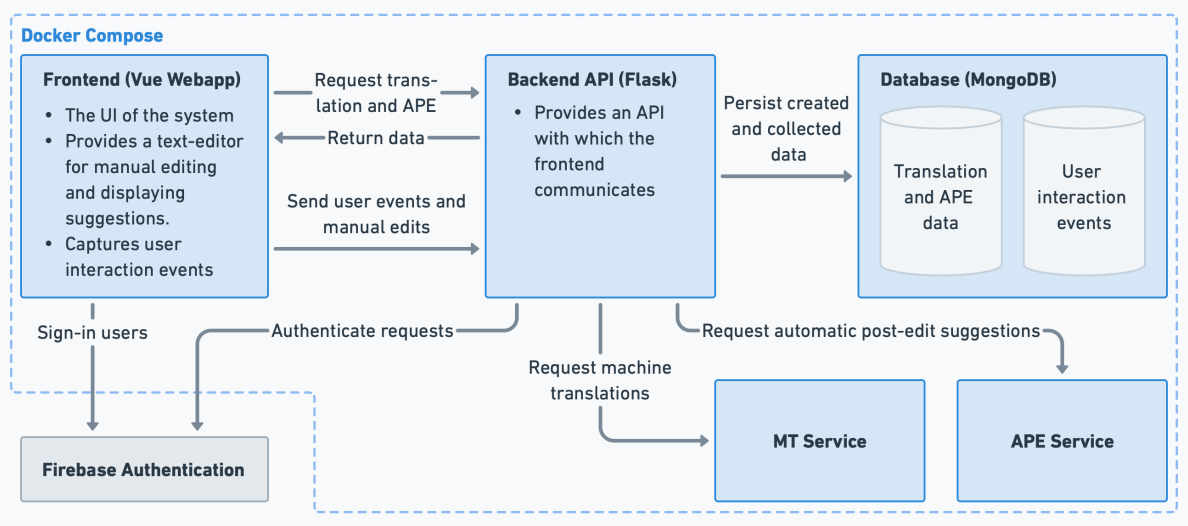

Figure 1: Overview of the OpenTIPE system architecture, which follows a microservice pattern, and the communication between the different components. Each blue box in the diagram represents a separate Docker container.

3 The OpenTIPE Framework

OpenTIPE implements a microservice architecture, consisting of independent services that are orchestrated with Docker Compose.[6] The advantages of this architecture are twofold. First, it enables separation of concerns. That is, each service implements its own function and runtime environment, reducing code complexity and errors. Second, its flexibility, as services can be easily replaced.

The main components of the OpenTIPE implementation are the frontend, which provides the graphical user interface to translate and post-edit texts, and the backend services: the backend API, MT and APE services, and data storage. Additionally, OpenTIPE supports user authentication with Firebase.[7][8] Figure 1 illustrates the overall architecture of OpenTIPE and the interaction between the different services. In the following, we describe the technical details of each component.

3.1 Frontend

The frontend implements the user interface of OpenTIPE, which currently consists of two main views: (1) the translation and (2) post-editing view (Figure 2). The translation view allows the user to add the text to translate and select different translation options. In particular, the user can choose among the available source and target languages, and define the translation of specific terms (see Section 3.4 for more details on the use of lexical constraints). In the post-editing view, the user can edit the automatic translation with the support of the post-editing suggestions from the APE model.

To facilitate its deployment, we adopt a web-based design. More specifically, we use VueJS, a lightweight JavaScript framework, and build the frontend as a single-page application.[9] This implementation runs entirely in the browser and decouples the backend business logic from the user interface, enhancing separation of concerns. To obtain the translations and the automatic post-editing suggestions, the frontend application communicates with the backend API (see Section 3.2).

An important feature of the user interface is its rich-text editor, which is implemented using the Tiptap framework.[10] The main strength of the Tiptap framework is its extensibility, allowing us to easily customise and add additional features. We write the entire frontend application in TypeScript[11] and host it statically using NGINX[12] in its corresponding docker container.

Document-level Post-editing Computer-assisted translation technologies typically organise the translation task in individual sentences. The human translator then addresses the document sentences one at a time, which helps to speed up the translation process. However, prior work reported that errors in current high-quality MT systems are harder

[6] https://docs.docker.com/compose/
[7] https://firebase.google.com/docs/auth
[8] Authentication can be entirely disabled if necessary.
[9] https://vuejs.org
[10] https://tiptap.dev
[11] https://www.typescriptlang.org
[12] https://www.nginx.com

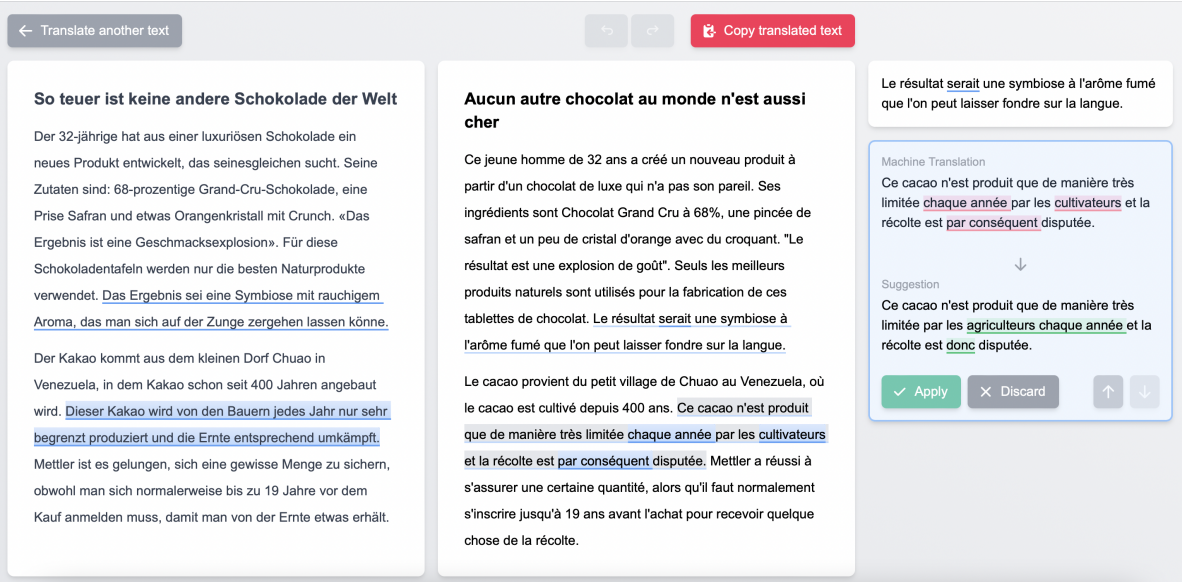

Figure 2: Post-editing view of OpenTIPE. The view shows the source text on the left, the post-editing area in the middle, which initially contains the MT output, and the APE suggestions on the right side. The user can discard or apply the suggestions to the revised text. The interface implements highlighting features to identify aligned source-target sentences and the differences between the translation and the corresponding APE suggestions. Users can click on the 'Copy translated text' button to save the current status of the revised translation locally.

to spot when focusing on isolated sentences instead of the whole document (Läubli et al., 2018). Additionally, a professional translator, who took part in our observation-based study (see details in Section 5), confirmed us the importance of document-level post-editing. We therefore display the entire document without segmentation in the post-editing view and offer highlighting features, which help the user to identify aligned sentences in the source and the target (see Figure 2). Nevertheless, the backend deals with the source, translation, and post-edited texts as independent sentences,[13] so it is possible to modify the user interface and display separate pairs of source-target sentences instead.

3.2 Backend API

The backend API acts as bridge between the frontend and the backend services (i.e. the database, the MT and APE models), defining the necessary endpoints to allow them to communicate and share resources. Specifically, it first provides the frontend with the automatic translations and post-editing suggestions from the MT and APE services, respectively (Section 3.4). Second, it stores the resulting post-editing metadata such as human post-edits and user interactions to the database (Section 3.3).

The API is based on Flask, a framework designed to build lightweight web applications,[14] and implemented in Python.[15] It is the only backend service that is accessible from the internet. Thus, if authentication is enabled, it connects to Firebase to validate and authenticate the incoming requests.

3.3 Data Storage

The data storage is based on MongoDB, a popular document-oriented NoSQL database.[16] The main strengths of MongoDB are its high performance and flexibility. In contrast to relational databases, MongoDB can support different data structures. Therefore, researchers can easily extend and modify the current implementation to fulfill their needs.

We divide the storage of the collected data, that is, the post-editing metadata and the logging of the user interactions, into two logical databases. Section 4 describes the data collection and its representation in more detail.

3.4 Translation and Post-editing Models

The OpenTIPE framework uses MT and APE models to obtain automatic translations and post-editing suggestions, respectively. We build these models in independent Docker containers, so they can be easily replaced. The current implementation of the MT

[13] We refer the reader to Section 4 for more details on the format and structure of the data.

[14] https://flask.palletsprojects.com/en/2.2.x/

[15] https://www.python.org

[16] https://www.mongodb.com

mtText	apeText	hpeText	apeAccepted
The Bremen town musicians	Town musicians of Bremen	Town musicians of Bremen	true
The Bremen town musicians	The Bremen town musicians	Town musicians of Bremen	false
Town musicians of Bremen	Town musicians of Bremen	Town musicians of Bremen	false
The Bremen town musicians	Town musicians of Bremen	The town musicians of Bremen	false
The Bremen town musicians	Town musicians of Bremen	The Bremen town musicians	false

Table 1: Examples of different values of the $textSegments$ properties for a single sentence object ($srcText$: 'Die Bremer Stadtmusikanten'). If there is no automatic suggestion (i.e. second and third rows), $apeText$ contains the $mtText$. The value of $hpeText$ is the final version of the sentence even if there is no correction.

service supports the use of the DeepL API[17] and Huggingface[18] or Fairseq Neural MT models.[19]

APE and Lexical Constraints In this post-editing environment, we consider the MT model as a black box and the improvements should be applied to the APE model. As an example, we release a simple APE implementation, an encoder-decoder architecture as in Correia and Martins (2019). In contrast to multi-source architectures, Correia and Martins (2019) use a single encoder whose input is the concatenated source and MT translation.[20]

Since post-editors are often required to use translation dictionaries, we extend the APE implementation to allow lexical constraints. That is, we can enforce the APE model to use specific translations for particular terms. To do so, we follow the approach described in Bergmanis and Pinnis (2021), which also handles the generation of the correct morphological inflection in the APE output.[21] This is specially important when translating into an inflected language, such as French. The approach augments the APE training data with the lemma of nouns and verbs such that the model learns to *copy and inflect*. For example, given the source text 'the improvement', we would augment it with the noun lemma in the target language (e.g. 'the improvement *retouche*'). As in Bergmanis and Pinnis (2021), we use the pre-trained Stanza models for lemmatization and part-of-speech tagging.[22] To define the translation constraints, the user can provide them in a file or introduce manual entries in the dictionary view of the user interface. During inference, we only augment those terms in the source for which the user specified a translation.

[17] https://www.deepl.com/docs-api

[18] https://huggingface.co/models

[19] https://github.com/facebookresearch/fairseq/blob/main/examples/translation/README.md

[20] https://github.com/deep-spin/OpenNMT-APE

[21] Jon et al. (2021) propose a similar approach to enforce lexical constraints and generate the corresponding inflection.

[22] https://github.com/stanfordnlp/stanza

key	description
$srcLang$	Language code of the source text.
$trgLang$	Language code of the target text.
$userDict$	Array of the manual translation entries defined by the user in the user interface.
$selectedDicts$	Array of the predefined dictionaries selected by the user.
$textSegments$	Array of sentence objects. Each sentence object contains the corresponding values for the source sentence ($srcText$), MT translation ($mtText$), automatic post-editing suggestion ($apeText$), revised version ($hpeText$), and a boolean indicating whether the automatic suggestion was accepted ($apeAccepted$).

Table 2: Description of the JSON object properties that define a post-edited document. See examples of sentence objects from $textSegments$ in Table 1.

4 Data Collection and Representation

Human post-edited data is a valuable resource to improve APE models. However, this is a scarce resource and researchers are often dependent on synthetic datasets. Therefore, the collection of human data is an important aspect of this IPE environment. Researchers can then leverage the data to implement human-in-the-loop approaches and assess the performance of different APE settings. Our implementation of OpenTIPE collects (1) human post-edited translations and (2) user interactions with the post-editing environment and it stores them as JSON objects in the MongoDB database (Section 3.3). The rest of this section describes the representation of these data in more detail.

Human Post-edited Translations We collect the human corrections together with additional relevant information, such as the corresponding source, MT output and use of translation dictionaries. OpenTIPE deals with the data at sentence-level, aligning sentence quartets of source, MT translation, APE suggestion, and proofread version. The

data is represented as a JSON object that defines the properties listed in Table 2. OpenTIPE captures and stores this data in the database when the user triggers the end of a revised version. That is, when the user copies the post-edited text using the copy keyboard shortcut or clicks on the 'Copy translated text' button of the user interface (see Figure 2). Note that different revisions of the same source text can be stored at different times.

User Interaction Logging We also record the user interactions with OpenTIPE in our database. This data can be used to evaluate different APE settings and the user experience with the editor. We currently log five types of events: $IdleEvent$, $ActiveEvent$, $AcceptEvent$, $RejectEvent$, and $CopyEvent$ together with the timestamp and user identifier.[23] The pair of events $IdleEvent$ and $ActiveEvent$ indicate the time intervals with user activity and inactivity. Specifically, we record an $IdleEvent$ when the user does not interact with the interface for a minute and an $ActiveEvent$ with any interaction after being idle (e.g. mouse click, scrolling). The event types $AcceptEvent$ and $RejectEvent$ are triggered when the user applies or discards automatic suggestions, respectively. Finally, $CopyEvent$ indicates that the user copied a post-editing revision locally.

5 Usability Study

We perform a user study to assess the usability of the OpenTIPE user interface. In particular, we conduct a controlled observation with a professional translator and a survey-based assessment with eight non-professional translators. The latter are academics between 21 and 30 years old (62.5% are male and 37.5% female), who indicated that they frequently use translation services, such as DeepL[24] and Google Translate.[25] While the observation-based setting allows us to get insights on the interactions of an expert with the tool, the survey-based assessment gives us a general subjective view of the user interface usability.

In both settings, all participants saw the interface of OpenTIPE for the first time during the study. We start the study explaining its purpose to the participants. In the observation-based setting, the professional translator is aware that he is being observed during the process.[26] We then provide them with a text to translate and the following instructions:

1. Translate the provided text using OpenTIPE.
2. Improve the automatic translation. For example, (a) apply automatic suggestions where needed or (b) rephrase the first sentence and split it in two sentences.
3. Save the final translation locally.

Furthermore, we ask the participants of the survey-based setting to fill in a questionnaire. The questionnaire consists of a set of questions as defined in the System Usability Scale (SUS) (Brooke, 1996) and three additional qualitative questions about what they liked the most and what features they think are missing or could be improved.

The OpenTIPE user interface obtained an average SUS score of 90, being 85 the lowest among the participants.[27] These results indicate that all participants evaluated the interface as excellent (Bangor et al., 2008). This is also confirmed with the answers to the qualitative questions. In fact, most of them stated that what they liked the most about the interface was its simplicity. Similarly, we observed that the professional translator used the interface as expected and could perform all tasks effortlessly.

6 Conclusion

We presented OpenTIPE, the first interactive framework that aims at supporting the research of human-centered approaches for post-editing. In contrast to research in machine translation, human-computer interaction has been only simulated for the post-editing task, since there was no interactive environment available for this purpose. OpenTIPE follows a microservice architecture such that it can be easily extended and adapted to other models or features. Additionally, it collects human post-editing data and the user interactions with the interface. These data are key to implement human-in-the-loop approaches that learn from human corrections over time. We expect this work to foster future research on interactive approaches that enhance the performance of the post-editing process. We are excited to explore this direction in future work.

[23] The logging can be easily extended with new event types.
[24] https://www.deepl.com/translator
[25] https://translate.google.com
[26] Two authors of this paper participated as observers.
[27] SUS scores have a range of 0 to 100 and a score over 68 is considered above average.

Ethics Statement

Usability Study We recruited the participants for our usability study on a voluntary basis and informed them of the goals and scope. Furthermore, we collected the data anonymously, such that no conclusion can be drawn about any participant. The usability study obtained the ethical approval (EK-2023-N-35) from the Ethics Commission of ETH Zurich university.

Translation and Post-editing Models We do not expect additional ethical concerns besides the already documented on natural language generator systems (Smiley et al., 2017; Kreps et al., 2022).

Potential Misuse Users could write undesired text (e.g. hateful or offensive comments) as post-edited text. As a result, the stored data could be used to train a model to generate texts that replicate this harmful behaviour. To mitigate this issue, we strongly recommend to activate the user authentication, so the framework is only accessible to trustworthy users. Additionally, researchers should periodically verify the data to filter those instances either manually or automatically, using a model to identify hallucinations in the text as in Su et al. (2022). Since bias can be present in the APE output, human-in-the-loop approaches can amplify this bias if the users heavily rely on the APE suggestions. Therefore, researchers should also debias the data regularly, for example, using existing tools such as AdaTest (Ribeiro and Lundberg, 2022).

Acknowledgements

This project is supported by Ringier, TX Group, NZZ, SRG, VSM, viscom, and the ETH Zurich Foundation.

References

Farhad Akhbardeh, Arkady Arkhangorodsky, Magdalena Biesialska, Ondřej Bojar, Rajen Chatterjee, Vishrav Chaudhary, Marta R. Costa-jussa, Cristina España-Bonet, Angela Fan, Christian Federmann, Markus Freitag, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Leonie Harter, Kenneth Heafield, Christopher Homan, Matthias Huck, Kwabena Amponsah-Kaakyire, Jungo Kasai, Daniel Khashabi, Kevin Knight, Tom Kocmi, Philipp Koehn, Nicholas Lourie, Christof Monz, Makoto Morishita, Masaaki Nagata, Ajay Nagesh, Toshiaki Nakazawa, Matteo Negri, Santanu Pal, Allahsera Auguste Tapo, Marco Turchi, Valentin Vydrin, and Marcos Zampieri. 2021. Findings of the 2021 conference on machine translation (WMT21). In *Proceedings of the Sixth Conference on Machine Translation*, pages 1–88, Online. Association for Computational Linguistics.

Vicent Alabau, Ragnar Bonk, Christian Buck, Michael Carl, Francisco Casacuberta, Mercedes García-Martínez, Jesús González-Rubio, Philipp Koehn, Luis A. Leiva, Bartolomé Mesa-Lao, Daniel Ortiz, Herve Saint-Amand, Germán Sanchis-Trilles, and Chara Tsoukala. 2013. CASMACAT: An open source workbench for advanced computer aided translation. *Prague Bulletin of Mathematical Linguistics*, 100:101–112.

Aaron Bangor, Philip T. Kortum, and James T. Miller. 2008. An empirical evaluation of the system usability scale. *International Journal of Human–Computer Interaction*, 24(6):574–594.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: a case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 257–267, Austin, Texas. Association for Computational Linguistics.

Toms Bergmanis and Mārcis Pinnis. 2021. Facilitating terminology translation with target lemma annotations. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3105–3111, Online. Association for Computational Linguistics.

Ondřej Bojar, Jindřich Helcl, Tom Kocmi, Jindřich Libovický, and Tomáš Musil. 2017. Results of the WMT17 neural MT training task. In *Proceedings of the Second Conference on Machine Translation*, pages 525–533, Copenhagen, Denmark. Association for Computational Linguistics.

John Brooke. 1996. *SUS: A quick and dirty usability scale*, volume 189, pages 4–7. Taylor & Francis.

Rajen Chatterjee, Gebremedhen Gebremelak, Matteo Negri, and Marco Turchi. 2017. Online automatic post-editing for MT in a multi-domain translation environment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 525–535, Valencia, Spain. Association for Computational Linguistics.

Rajen Chatterjee, Marion Weller, Matteo Negri, and Marco Turchi. 2015. Exploring the planet of the APEs: a comparative study of state-of-the-art methods for MT automatic post-editing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 156–161, Beijing, China. Association for Computational Linguistics.

Gonçalo M. Correia and André F. T. Martins. 2019. A simple and effective approach to automatic post-editing with transfer learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3050–3056, Florence, Italy. Association for Computational Linguistics.

Félix do Carmo, Dimitar Shterionov, Joss Moorkens, Joachim Wagner, Murhaf Hossari, Eric Paquin, Dag Schmidtke, Declan Groves, and Andy Way. 2020. A review of the state-of-the-art in automatic post-editing. *Machine Translation*, pages 1–43.

Marie Escribe and Ruslan Mitkov. 2021. Interactive models for post-editing. In *Proceedings of the Translation and Interpreting Technology Online Conference*, pages 167–173, Held Online. INCOMA Ltd.

José Esteban, José Lorenzo, Antonio S. Valderrábanos, and Guy Lapalme. 2004. TransType2 - an innovative computer-assisted translation system. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 94–97, Barcelona, Spain. Association for Computational Linguistics.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018. Search engine guided neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Josef Jon, João Paulo Aires, Dusan Varis, and Ondřej Bojar. 2021. End-to-end lexically constrained machine translation for morphologically rich languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4019–4033, Online. Association for Computational Linguistics.

Marcin Junczys-Dowmunt. 2018. Are we experiencing the golden age of automatic post-editing? In *Proceedings of the AMTA 2018 Workshop on Translation Quality Estimation and Automatic Post-Editing*, pages 144–206, Boston, MA. Association for Machine Translation in the Americas.

Rebecca Knowles and Philipp Koehn. 2016. Neural interactive translation prediction. In *Conferences of the Association for Machine Translation in the Americas: MT Researchers' Track*, pages 107–120, Austin, TX, USA. The Association for Machine Translation in the Americas.

Sachith Sri Ram Kothur, Rebecca Knowles, and Philipp Koehn. 2018. Document-level adaptation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 64–73, Melbourne, Australia. Association for Computational Linguistics.

Sarah Kreps, R. Miles McCain, and Miles Brundage. 2022. All the news that's fit to fabricate: AI-generated text as a tool of media misinformation. *Journal of Experimental Political Science*, 9(1):104–117.

Samuel Läubli, Rico Sennrich, and Martin Volk. 2018. Has machine translation achieved human parity? a case for document-level evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4791–4796, Brussels, Belgium. Association for Computational Linguistics.

Matteo Negri, Marco Turchi, Nicola Bertoldi, and Marcello Federico. 2018a. Online neural automatic post-editing for neural machine translation. In *Fifth Italian Conference on Computational Linguistics (CLiC-it 2018)*.

Matteo Negri, Marco Turchi, Rajen Chatterjee, and Nicola Bertoldi. 2018b. ESCAPE: a large-scale synthetic corpus for automatic post-editing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Daniel Ortiz-Martínez and Francisco Casacuberta. 2014. The new thot toolkit for fully-automatic and interactive statistical machine translation. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 45–48, Gothenburg, Sweden. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Marco Tulio Ribeiro and Scott Lundberg. 2022. Adaptive testing and debugging of NLP models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3253–3267, Dublin, Ireland. Association for Computational Linguistics.

Sebastin Santy, Sandipan Dandapat, Monojit Choudhury, and Kalika Bali. 2019. INMT: Interactive neural machine translation prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 103–108, Hong Kong, China. Association for Computational Linguistics.

Michel Simard and George Foster. 2013. PEPr: Post-edit propagation using phrase-based statistical machine translation. In *Proceedings of Machine Translation Summit XIV: Papers*, Nice, France.

Charese Smiley, Frank Schilder, Vassilis Plachouras, and Jochen L. Leidner. 2017. Say the right thing right: Ethics issues in natural language generation systems. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages

103–108, Valencia, Spain. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas.

Dan Su, Xiaoguang Li, Jindi Zhang, Lifeng Shang, Xin Jiang, Qun Liu, and Pascale Fung. 2022. Read before generate! faithful long form question answering with machine reading. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 744–756, Dublin, Ireland. Association for Computational Linguistics.

Antonio Toral. 2020. Reassessing claims of human parity and super-human performance in machine translation at WMT 2019. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 185–194, Lisboa, Portugal. European Association for Machine Translation.

Dongqi Wang, Haoran Wei, Zhirui Zhang, Shujian Huang, Jun Xie, Weihua Luo, and Jiajun Chen. 2021. Non-parametric online learning from human feedback for neural machine translation. *CoRR*, abs/2109.11136.